

FFFFFFFFFFFFFFFF	111	111	AAAAAAAAAA	
FFFFFFFFFFFFFFFF	111	111	AAAAAAAAAA	
FFFFFFFFFFFFFFFF	111	111	AAAAAAAAAA	
FFF	111111	111111	AAA	AAA
FFF	111111	111111	AAA	AAA
FFF	111111	111111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFFFFFFFFFFFFF	111	111	AAA	AAA
FFF	111	111	AAAAAAAAAAAAAAAA	AAA
FFF	111	111	AAAAAAAAAAAAAAAA	AAA
FFF	111	111	AAAAAAAAAAAAAAAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	111	111	AAA	AAA
FFF	1111111111	1111111111	AAA	AAA
FFF	1111111111	1111111111	AAA	AAA
FFF	1111111111	1111111111	AAA	AAA

A large grid of 100 characters (A, C, E, S) arranged in a 10x10 pattern. The characters are distributed as follows:

- Row 1: AAAAAA, CCCCCCCC, CCCCCCCC, EEEEEEEEEE, EEEEEEEEEE, SSSSSSSS, SSSSSSSS
- Row 2: AA, AA, CC, CC, CC, SS, SS
- Row 3: AA, AA, CC, CC, CC, SS, SS
- Row 4: AA, AA, CC, CC, CC, SS, SS
- Row 5: AA, AA, CC, CC, CC, SS, SS
- Row 6: AA, AA, CC, CC, CC, SS, SS
- Row 7: AAAAAAAAAA, CC, CC, CC, CC, SS, SS
- Row 8: AAAAAAAAAA, CC, CC, CC, CC, SS, SS
- Row 9: AA, AA, CC, CC, CC, SS, SS
- Row 10: AA, AA, CCCCCCCC, CCCCCCCC, EEEEEEEEEE, SSSSSSSS, SSSSSSSS

The last row is partially cut off by ellipses on the right side.

```

LL          IIIIII          SSSSSSSS
LL          IIIIII          SSSSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SSSSSS
LL          II             SSSSSS
LL          II             SS
LL          II             SS
LL          II             SS
LL          II             SS
LLLLLLLLLLLL IIIIII          SSSSSSSS
LLLLLLLLLLLL IIIIII          SSSSSSSS

```

```
1 0001 0 MODULE ACCESS (
2 0002 0 LANGUAGE (BLISS32),
3 0003 0 IDENT = 'V04-000',
4 0004 0 ) =
5 0005 1 BEGIN
6 0006 1
7 0007 1
8 0008 1 *****
9 0009 1 *
10 0010 1 * COPYRIGHT (c) 1978, 1980, 1982, 1984 BY
11 0011 1 * DIGITAL EQUIPMENT CORPORATION, MAYNARD, MASSACHUSETTS.
12 0012 1 * ALL RIGHTS RESERVED.
13 0013 1 *
14 0014 1 * THIS SOFTWARE IS FURNISHED UNDER A LICENSE AND MAY BE USED AND COPIED
15 0015 1 * ONLY IN ACCORDANCE WITH THE TERMS OF SUCH LICENSE AND WITH THE
16 0016 1 * INCLUSION OF THE ABOVE COPYRIGHT NOTICE. THIS SOFTWARE OR ANY OTHER
17 0017 1 * COPIES THEREOF MAY NOT BE PROVIDED OR OTHERWISE MADE AVAILABLE TO ANY
18 0018 1 * OTHER PERSON. NO TITLE TO AND OWNERSHIP OF THE SOFTWARE IS HEREBY
19 0019 1 * TRANSFERRED.
20 0020 1 *
21 0021 1 * THE INFORMATION IN THIS SOFTWARE IS SUBJECT TO CHANGE WITHOUT NOTICE
22 0022 1 * AND SHOULD NOT BE CONSTRUED AS A COMMITMENT BY DIGITAL EQUIPMENT
23 0023 1 * CORPORATION.
24 0024 1 *
25 0025 1 * DIGITAL ASSUMES NO RESPONSIBILITY FOR THE USE OR RELIABILITY OF ITS
26 0026 1 * SOFTWARE ON EQUIPMENT WHICH IS NOT SUPPLIED BY DIGITAL.
27 0027 1 *
28 0028 1 *
29 0029 1 *****
30 0030 1
31 0031 1 ++
32 0032 1
33 0033 1 FACILITY: F11ACP Structure Level 1
34 0034 1
35 0035 1 ABSTRACT:
36 0036 1
37 0037 1 This is the main processing routine for the ACCESS function.
38 0038 1
39 0039 1 ENVIRONMENT:
40 0040 1
41 0041 1 STARLET operating system, including privileged system services
42 0042 1 and internal exec routines.
43 0043 1
44 0044 1 --
45 0045 1
46 0046 1
47 0047 1 AUTHOR: Andrew C. Goldstein, CREATION DATE: 20-Dec-1976 15:43
48 0048 1
49 0049 1 MODIFIED BY:
50 0050 1
51 0051 1 V03-003 LMP0154 L. Mark Pilant, 19-Sep-1983 12:36
52 0052 1 Return to the primary header after reading the attributes.
53 0053 1
54 0054 1 V03-002 LMP0023 L. Mark Pilant, 8-Apr-1982 10:40
55 0055 1 If there is only one FCB, don't call REMAP_FILE but still
56 0056 1 set COMPLETE in the window.
57 0057 1
```


ACCESS
V04-000

N 2
16-Sep-1984 00:46:05 VAX-11 Bliss-32 V4.0-742 Page 2
14-Sep-1984 12:29:18 DISK\$VMSMASTER:[F11A.SRC]ACCESS.B32;1 (1)

```

58 0058 1 |
59 0059 1 |
60 0060 1 |
61 0061 1 |
62 0062 1 |
63 0063 1 |
64 0064 1 |
65 0065 1 |
66 0066 1 |
67 0067 1 |
68 0068 1 |
69 0069 1 |
70 0070 1 |
71 0071 1 |
72 0072 1 |
73 0073 1 |
74 0074 1 |
75 0389 1 |
76 0390 1 |
77 0391 1 |
78 0392 1 |
79 0393 1 |
80 0394 1 |

V03-001 LMP0018 L. Mark Pilant, 31-Mar-1982 12:00
Modify to use a local copy of the window complete flag.

V02-005 LMP0005 L. Mark Pilant, 29-Dec-1981 12:35
Added byte limit quota check on window creation and
Cathedral window support.

V02-004 ACG0225 Andrew C. Goldstein, 24-Nov-1981 17:39
Add NOLOCK support

V02-003 ACG0167 Andrew C. Goldstein, 7-May-1980 18:47
Previous revision history moved to F11A.REV

**

LIBRARY 'SYSS$LIBRARY:LIB.L32';
REQUIRE 'SRCS:FCPDEF.B32';

FORWARD ROUTINE
ACCESS, ! main access function processing
CHECK FIND, ! conditional directory search
HANDLER; ! conditional handler to catch error exit
```

ACC
V04

: R

```
82 0395 1 GLOBAL ROUTINE ACCESS =
83 0396 1
84 0397 1 ++
85 0398 1
86 0399 1 FUNCTIONAL DESCRIPTION:
87 0400 1
88 0401 1 This is the main processing routine for the ACCESS function.
89 0402 1
90 0403 1 CALLING SEQUENCE:
91 0404 1 ACCESS ()
92 0405 1
93 0406 1 INPUT PARAMETERS:
94 0407 1 NONE
95 0408 1
96 0409 1 IMPLICIT INPUTS:
97 0410 1 CURRENT_VCB: VCB of volume
98 0411 1 IO_PACKET: address of I/O request packet
99 0412 1
100 0413 1 OUTPUT PARAMETERS:
101 0414 1 NONE
102 0415 1
103 0416 1 IMPLICIT OUTPUTS:
104 0417 1 PRIMARY_FCB: FCB of file
105 0418 1 CURRENT_WINDOW: address of file window
106 0419 1 USER_STATUS: I/O status block to return to user
107 0420 1
108 0421 1 ROUTINE VALUE:
109 0422 1 NONE
110 0423 1
111 0424 1 SIDE EFFECTS:
112 0425 1 FCB & window created
113 0426 1
114 0427 1 --
115 0428 1
116 0429 2 BEGIN
117 0430 2
118 0431 2 LOCAL
119 0432 2 FCB_CREATED, : flag indicating new FCB created
120 0433 2 PACKET : REF BBLOCK, : address of I/O packet
121 0434 2 ABD : REF BBLOCKVECTOR [,ABD$C_LENGTH], :
122 0435 2 : buffer descriptors
123 0436 2 FIB : REF BBLOCK, : file identification block
124 0437 2 FCB : REF BBLOCK, : FCB address
125 0438 2 HEADER : REF BBLOCK, : address of file header
126 0439 2 NEW_HEADER : REF BBLOCK, : address of extension header
127 0440 2 FUNCTION : BLOCK [1]; : function code qualifiers
128 0441 2
129 0442 2 EXTERNAL
130 0443 2 USER_STATUS : VECTOR, : I/O status block for user
131 0444 2 CURRENT_VCB : REF BBLOCK, : VCB of volume
132 0445 2 PRIMARY_FCB : REF BBLOCK, : FCB of file
133 0446 2 CURRENT_WINDOW : REF BBLOCK, : window for file
134 0447 2 IO_PACKET : REF BBLOCK, : I/O request packet
135 0448 2 CLEANUP_FLAGS : BITVECTOR, : cleanup action flags
136 0449 2 FILE_HEADER : REF BBLOCK; : Address of current file header
137 0450 2
138 0451 2 EXTERNAL ROUTINE
```



```
139 0452 2 GET FIB, | get FIB for operation
140 0453 2 FIND, | find file in directory
141 0454 2 CREATE, | create file
142 0455 2 SEARCH_FCB, | search FCB list
143 0456 2 READ_HEADER, | read file header
144 0457 2 NEXT_HEADER, | read extension file header
145 0458 2 CREATE_FCB, | create an FCB
146 0459 2 CHECK_PROTECT, | check file protection
147 0460 2 CREATE_WINDOW, | create a window
148 0461 2 MAKE_ACCESS, | complete the access
149 0462 2 MAKE_EXTFCB, | create and link extension FCB
150 0463 2 FLUSH_FID, | flush a file from the buffer pool
151 0464 2 UPDATE_FCB, | update attributes in FCB
152 0465 2 READ_ATTRIB, | read file attributes
153 0466 2 REMAP_FILE, | remap the file completely
154 0467 2 MARK_COMPLETE; | mark file complete
155 0468
156 0469
157 0470 2 ! Enable the deaccess cleanup if an access is taking place.
158 0471 2 !
159 0472 2
160 0473 2 PACKET = .IO PACKET;
161 0474 2 FUNCTION = .PACKET[IRPSW_FUNC];
162 0475 2 IF .FUNCTION[IOSV_ACCESS]
163 0476 2 THEN
164 0477 2 BEGIN
165 0478 2 CLEANUP_FLAGS[CLF_ZCHANNEL] = 1;
166 0479 2 CLEANUP_FLAGS[CLF_DELWINDOW] = 1;
167 0480 2 END;
168 0481 2
169 0482 2 ! Set up pointers to interesting control blocks.
170 0483 2 !
171 0484 2
172 0485 2 ! pointer to buffer descriptors
173 0486 2 ABD = .BBLOCK [.PACKET[IRPSL_SVAPTE], AIB$L_DESCRIPTOR];
174 0487 2 FIB = GET_FIB (.ABD); | pointer to FIB
175 0488 2
176 0489 2 ! Do directory processing, if any. For a normal access, do the directory
177 0490 2 ! lookup if a directory ID is present. If this is a conditional create, do
178 0491 2 ! the lookup and turn the function into a create if the lookup fails
179 0492 2 ! with a file not found. Conditional create on spool devices always become
180 0493 2 ! creates.
181 0494 2 !
182 0495 2
183 0496 2 IF .FUNCTION[IOSV_CREATE]
184 0497 2 THEN
185 0498 2 BEGIN
186 0499 2 IF .CLEANUP_FLAGS[CLF_SPOOLFILE]
187 0500 2 OR (
188 0501 2 IF .FIB[FIB$W_DID_NUM] NEQ 0
189 0502 2 THEN NOT CHECK_FIND (.ABD, .FIB)
190 0503 2 ELSE 1
191 0504 2 )
192 0505 2 THEN
193 0506 2 BEGIN
194 0507 2 USER_STATUS[0] = SSS_CREATED;
195 0508 2 RETURN CREATE ();
```

```
196 0509 3 END;
197 0510 3 END
198 0511 3
199 0512 3 ELSE
200 0513 3 IF .FIB[FIB$W_DID_NUM] NEQ 0
201 0514 3 THEN FIND (.ABD, .FIB, 0);
202 0515 3
203 0516 3 ! If there is a file open on the channel, check the file ID returned by the
204 0517 3 FIND against the file ID that is open. If they are different, drop the FCB
205 0518 3 and window addresses on the floor.
206 0519 3
207 0520 3
208 0521 3 IF .PRIMARY_FCB NEQ 0
209 0522 3 THEN
210 0523 3 IF .PRIMARY_FCB[FCB$W_FID_NUM] NEQ .FIB[FIB$W_FID_NUM]
211 0524 3 OR .PRIMARY_FCB[FCB$W_FID_RVN] NEQ .FIB[FIB$W_FID_RVN]
212 0525 3 THEN
213 0526 3 BEGIN
214 0527 3 PRIMARY_FCB = 0;
215 0528 3 CURRENT_WINDOW = 0;
216 0529 3 END;
217 0530 3
218 0531 3 ! If this is a find only, exit now to avoid an extraneous read of the
219 0532 3 file header.
220 0533 3
221 0534 3
222 0535 3 IF NOT .FUNCTION[IO$V_ACCESS] ! if no access
223 0536 3 AND .PACKET[IRP$W_BCNT] LEQ ABD$C_ATTRIB ! and no attribute list
224 0537 3 THEN RETURN 1; ! all done
225 0538 3
226 0539 3 ! Find the FCB of the file, if one exists. then read the file
227 0540 3 header. If there is no FCB, create one.
228 0541 3
229 0542 3
230 0543 3 FCB = SEARCH_FCB (FIB[FIB$W_FID]);
231 0544 3 HEADER = READ_HEADER (FIB[FIB$W_FID], .FCB);
232 0545 3
233 0546 3 ! If the file is marked for delete and is not accessed by this user, and
234 0547 3 the accessor is not the system, deny its existence.
235 0548 3
236 0549 3
237 0550 3 IF .CURRENT_WINDOW EQL 0 AND .HEADER[FH1$V_MARKDEL]
238 0551 3 AND NOT .BBLOCK [BBLOCK [.PACKET[IRP$L_ARB], ARB$Q_PRIV], PRV$V_BYPASS]
239 0552 3 THEN ERR_EXIT (SS$NOSUCHFILE);
240 0553 3
241 0554 3 FCB_CREATED = 0;
242 0555 3 IF .FCB EQL 0
243 0556 3 THEN
244 0557 3 BEGIN
245 0558 3 FCB_CREATED = 1;
246 0559 3 FCB = KERNEL_CALL (CREATE_FCB, .HEADER);
247 0560 3 END;
248 0561 3 PRIMARY_FCB = .FCB; ! record FCB for external use
249 0562 3
250 0563 3 ! If access is requested, check for conflicts and file protection.
251 0564 3 then create a window and link everything up.
252 0565 3
```



```
253 0566 2
254 0567 2 IF .FUNCTION[IOSV_ACCESS]
255 0568 2 THEN
256 0569 2 BEGIN
257 0570 2 CHECK_PROTECT (.FIB[FIB$V_WRITE] OR .FIB[FIB$V_NOREAD], .HEADER, .FCB);
258 0571 2
259 0572 2 IF (.HEADER[FH1$V_LOCKED])
260 0573 2 AND NOT .BBLOCK [BBLOCK [.PACKET[IRP$L_ARB], ARB$Q_PRIV], PRV$V_BYPASS]
261 0574 2 THEN ERR_EXIT (SS$_FILELOCKED); ! file is deaccess locked
262 0575 2
263 0576 2 IF (.FCB[FCB$V_EXCL]
264 0577 2 AND NOT (.FIB[FIB$V_NOLOCK] AND .CLEANUP_FLAGS[CLF_SYSPRV]))
265 0578 2 OR (.FIB[FIB$V_NOREAD] AND .FCB[FCB$W_ACNT] NEQ 0)
266 0579 2 OR (.FIB[FIB$V_NOWRITE] AND .FCB[FCB$W_WCNT] NEQ 0)
267 0580 2 OR (.FIB[FIB$V_WRITE] AND .FCB[FCB$W_LCNT] NEQ 0)
268 0581 2 OR (.FCB[FCB$W_SEGN] NEQ 0 AND .FCB[FCB$W_ACNT] NEQ 0)
269 0582 2 THEN ERR_EXIT (SS$_ACCONFLICT); ! one of above access conflicts
270 0583 2
271 0584 2
272 P 0585 2 CURRENT_WINDOW = KERNEL_CALL (CREATE_WINDOW, .FIB[FIB$L_ACCTL],
273 0586 2 .FIB[FIB$B_WSIZE], .HEADER, .PACKET[IRP$L_PID], .FCB);
274 0587 2 IF .CURRENT_WINDOW EQL 0 THEN ERR_EXIT (SS$_EXBYTLM);
275 0588 2 KERNEL_CALL (MAKE_ACCESS, .FCB, .CURRENT_WINDOW, .ABD);
276 0589 2
277 0590 2 ! If the file looks like a directory file and it is being write accessed,
278 0591 2 ! flush it from the buffer pool to avoid retaining stale directory data.
279 0592 2
280 0593 2
281 0594 2 IF .FIB[FIB$V_WRITE]
282 0595 2 AND .BBLOCK [HEADER[FH1$W_RECATTR], FAT$B_RTYPE] EQL FAT$C_FIXED
283 0596 2 AND .BBLOCK [HEADER[FH1$W_RECATTR], FAT$W_RSIZE] EQL NMB$C_DIRENTRY
284 0597 2 THEN FLUSH_FID (FIB[FIB$W_FID]);
285 0598 2
286 0599 2 END; ! end of access processing
287 0600 2
288 0601 2 ! If the file is multi-header, read the extension headers and create
289 0602 2 ! extension FCB's as necessary. finally read back the primary header.
290 0603 2
291 0604 2
292 0605 2 IF .FCB_CREATED
293 0606 2 THEN
294 0607 2 BEGIN
295 0608 2 WHILE 1 DO
296 0609 2 BEGIN
297 0610 2 NEW_HEADER = NEXT_HEADER (.HEADER, .FCB);
298 0611 2 IF .NEW_HEADER EQL 0 THEN EXITLOOP;
299 0612 2 HEADER = .NEW_HEADER;
300 0613 2 IF .FUNCTION[IOSV_ACCESS]
301 0614 2 AND SEARCH_FCB (HEADER[FH1$W_FID]) NEQ 0
302 0615 2 THEN ERR_EXIT (SS$_ACCONFLICT);
303 0616 2 FCB = KERNEL_CALL (MAKE_EXTFCB, .HEADER, .FCB, .FUNCTION[IOSV_ACCESS]);
304 0617 2 END;
305 0618 2
306 0619 2 IF .FCB NEQ .PRIMARY_FCB
307 0620 2 THEN
308 0621 2 BEGIN
309 0622 2 FCB = .PRIMARY_FCB;
```



```
0623 4      HEADER = READ HEADER (0, .FCB);
0624 4      KERNEL_CALL (UPDATE_FCB, .HEADER);
0625 4      END;
0626 4
0627 4      ! Do read attributes if requested.
0628 4      !
0629 4
0630 4      IF .PACKET[IRPSW_BCNT] GTR ABD$C_ATTRIB
0631 4      THEN
0632 4      BEGIN
0633 4      IF .CURRENT_WINDOW EQL 0
0634 4      THEN CHECK_PROTECT (RDATT_ACCESS, .HEADER, .FCB);
0635 4      IF NOT KERNEL_CALL (READ_ATTRIB, .HEADER, .ABD) THEN ERR_EXIT ();
0636 4      HEADER = .FILE_HEADER;
0637 4      END;
0638 4
0639 4      ! If necessary map the file completely.
0640 4
0641 4      IF .FUNCTION[IOSV_ACCESS]
0642 4      THEN
0643 4      IF .CURRENT_WINDOW[WCB$V_CATHEDRAL]
0644 4      THEN
0645 4      IF .PRIMARY_FCB[FCB$SL_EXFCB] NEQ 0
0646 4      THEN REMAP_FILE();
0647 4      ELSE KERNEC_CALL (MARK_COMPLETE, .CURRENT_WINDOW);
0648 4
0649 4      RETURN 1;
0650 4
0651 4      ! end of routine ACCESS
0652 1 END;
```

```
.TITLE ACCESS
.IDENT \V04-000\
```

```
.EXTRN USER STATUS, CURRENT_VCB
.EXTRN PRIMARY_FCB, CURRENT_WINDOW
.EXTRN IO_PACKET, CLEANUP_FLAGS
.EXTRN FILE_HEADER, GET_FIB
.EXTRN FIND, CREATE, SEARCH_FCB
.EXTRN READ_HEADER, NEXT_HEADER
.EXTRN CREATE_FCB, CHECK_PROTECT
.EXTRN CREATE_WINDOW, MAKE_ACCESS
.EXTRN MAKE_EXTFCB, FLUSH_FID
.EXTRN UPDATE_FCB, READ_ATTRIB
.EXTRN REMAP_FILE, MARK_COMPLETE
.EXTRN SYSS$CMKRNL
```

```
.PSECT $CODE$,NOWRT,2
```

```
.ENTRY ACCESS, Save R2,R3,R4,R5,R6,R7,R8,R9,R10,- : 0395
R11
MOVAB CURRENT_WINDOW, R11
MOVAB PRIMARY_FCB, R10
MOVAB @SYSS$CMKRNL, R9
MOVL IO_PACKET, PACKET : 0473
MOVZWL 32(PACKET), FUNCTION : 0474
```

```
OFFC 00000
```

```
5B 0000G CF 9E 00002
5A 0000G CF 9E 00007
59 00000000G 9F 9E 0000C
55 0000G CF D0 00013
56 20 A5 3C 00018
```

07	0000G	56	0402	06	E1	0001C	BBC	#6, FUNCTION, 1\$	0475	
		CF		8F	A8	00020	BISW2	#1026, CLEANUP_FLAGS+2	0479	
		57	2C	B5	D0	00027	1\$:	MOVL	244(PACKET), ABD	0486
	0000G	CF		57	DD	0002B	PUSHL	ABD	0487	
		52		01	FB	0002D	CALLS	#1, GET_FIB		
				50	D0	00032	MOVL	R0, FIB		
				56	95	00035	TSTB	FUNCTION	0496	
			0000G	24	1B	00037	BGEQ	3\$		
				CF	95	00039	TSTB	CLEANUP_FLAGS	0499	
			0A	11	19	0003D	BLSS	2\$		
				A2	B5	0003F	TSTW	10(FIB)	0501	
				0C	13	00042	BEQL	2\$		
				52	DD	00044	PUSHL	FIB	0502	
				57	DD	00046	PUSHL	ABD		
	0000V	CF		02	FB	00048	CALLS	#2, CHECK_FIND		
		1D		50	E8	0004D	BLBS	R0, 4\$		
	0000G	CF	0619	8F	3C	00050	2\$:	MOVZWL	#1561, USER_STATUS	0507
	0000G	CF		00	FB	00057	CALLS	#0, CREATE	0508	
				04	00	0005C	RET			
			0A	A2	B5	0005D	3\$:	TSTW	10(FIB)	0513
				0B	13	00060	BEQL	4\$		
				7E	D4	00062	CLRL	-(SP)	0514	
				52	DD	00064	PUSHL	FIB		
				57	DD	00066	PUSHL	ABD		
	0000G	CF		03	FB	00068	CALLS	#3, FIND		
		50		6A	D0	0006D	4\$:	MOVL	PRIMARY_FCB, R0	0521
				12	13	00070	BEQL	6\$		
	04	A2	24	A0	B1	00072	CMPW	36(R0), 4(FIB)	0523	
				07	12	00077	BNEQ	5\$		
	08	A2	28	A0	B1	00079	CMPW	40(R0), 8(FIB)	0524	
				04	13	0007E	BEQL	6\$		
				6A	D4	00080	5\$:	CLRL	PRIMARY_FCB	0527
				6B	D4	00082	CLRL	CURRENT_WINDOW	0528	
09		56		06	E0	00084	6\$:	BBS	#6, FUNCTION, 7\$	0535
		05	32	A5	B1	00088	CMPW	50(PACKET), #5	0536	
				03	1A	0008C	BGTRU	7\$		
			01AF	31	0008E	BRW	30\$			
			04	A2	9F	00091	7\$:	PUSHAB	4(FIB)	0543
	0000G	CF		01	FB	00094	CALLS	#1, SEARCH_FCB		
		53		50	D0	00099	MOVL	R0, FCB		
				53	DD	0009C	PUSHL	FCB	0544	
			04	A2	9F	0009E	PUSHAB	4(FIB)		
	0000G	CF		02	FB	000A1	CALLS	#2, READ_HEADER		
		54		50	D0	000A6	MOVL	R0, HEADER		
				6B	D5	000A9	TSTL	CURRENT_WINDOW	0550	
				0F	12	000AB	BNEQ	8\$		
			0D	A4	95	000AD	TSTB	13(HEADER)		
				0A	18	000B0	BGEQ	8\$		
05	58	B5	0910	1D	E0	000B2	BBS	#29, 288(PACKET), 8\$	0551	
				8F	BF	000B7	CHMU	#2320	0552	
				04	00	000BB	RET			
				58	D4	000BC	8\$:	CLRL	FCB_CREATED	0554
				53	D5	000BE	TSTL	FCB	0555	
				13	12	000C0	BNEQ	9\$		
		58		01	D0	000C2	MOVL	#1, FCB_CREATED	0558	
				54	DD	000C5	PUSHL	HEADER	0559	
				01	DD	000C7	PUSHL	#1		

				69	0000G	SE DD 000C9	PUSHL SP		
				53		CF 9F 000CB	PUSHAB CREATE_FCB		
				6A		04 FB 000CF	CALLS #4, SYSSCMKRN		
	03			56		50 D0 000D2	MOVL R0, FCB		0561
						53 D0 000D5	MOVL FCB, PRIMARY_FCB		0567
						06 E0 000D8	BBS #6 FUNCTION, 10\$		
					00A7	31 000DC	BRW 20\$		
						53 DD 000DF	PUSHL FCB		0570
						54 DD 000E1	PUSHL HEADER		
						00 EF 000E3	EXTZV #0, #1 1(FIB), R0		
						0A EF 000E9	EXTZV #10, #1 (FIB), R1		
						51 C9 000EE	BISL3 R1, R0, -(SP)		
						03 CB 000F2	CALLS #3 CHECK PROTECT		
						06 E1 000F7	BBC #6 12(HEADER), 11\$		0572
						1D E0 000FC	BBS #29, @88(PACKET), 11\$		0573
					08A8	8F BF 00101	CHMU #2216		0574
						04 00105	RET		
						03 E1 00106	BBC #3 34(FCB), 14\$		0576
						14 E0 0010B	BBS #20, (FIB), 13\$		0577
					0098	31 0010F	BRW 22\$		
					0000G	CF E9 00112	BLBC CLEANUP_FLAGS+1, 12\$		
						0A E1 00117	BBC #10, (FIB), 15\$		0578
					1A	A3 B5 0011B	TSTW 26(FCB)		
						EF 12 0011E	BNEQ 12\$		
				05		62 E9 00120	BLBC (FIB), 16\$		0579
					1C	A3 B5 00123	TSTW 28(FCB)		
						E7 12 00126	BNEQ 12\$		
				05	01	A2 E9 00128	BLBC 1(FIB), 17\$		0580
					1E	A3 B5 0012C	TSTW 30(FCB)		
						79 12 0012F	BNEQ 22\$		
					2A	A3 B5 00131	TSTW 42(FCB)		0581
						05 13 00134	BEQL 18\$		
					1A	A3 B5 00136	TSTW 26(FCB)		
						6F 12 00139	BNEQ 22\$		
						53 DD 0013B	PUSHL FCB		0586
					0C	A5 DD 0013D	PUSHL 12(PACKET)		
						54 DD 00140	PUSHL HEADER		
				7E	03	A2 98 00142	CVTBL 3(FIB), -(SP)		
						62 DD 00146	PUSHL (FIB)		
						05 DD 00148	PUSHL #5		
						5E DD 0014A	PUSHL SP		
					0000G	CF 9F 0014C	PUSHAB CREATE_WINDOW		
						08 FB 00150	CALLS #8, SYSSCMKRN		
						50 D0 00153	MOVL R0 CURRENT_WINDOW		0587
						05 12 00156	BNEQ 19\$		
					2A14	8F BF 00158	CHMU #10772		
						04 0015C	RET		
						57 DD 0015D	PUSHL ABD		0588
						6B DD 0015F	PUSHL CURRENT_WINDOW		
						53 DD 00161	PUSHL FCB		
						03 DD 00163	PUSHL #3		
						5E DD 00165	PUSHL SP		
					0000G	CF 9F 00167	PUSHAB MAKE_ACCESS		
						06 FB 0016B	CALLS #6, SYSSCMKRN		
					01	A2 E9 0016E	BLBC 1(FIB), 20\$		0594
						A4 91 00172	CMPB 14(HEADER), #1		0595
						0E 12 00176	BNEQ 20\$		

		10	10	A4	B1	00178	CMPL	16(HEADER), #16	0596
				08	12	0017C	BNEQ	20\$	
			04	A2	9F	0017E	PUSHAB	4(F1B)	0597
	0000G	CF		01	FB	00181	CALLS	#1, FLUSH FID	
		60		58	E9	00186	BLBC	FCB_CREATED, 25\$	0605
				53	DD	00189	PUSHL	FCB	0610
				54	DD	0018B	PUSHL	HEADER	
	0000G	CF		02	FB	0018D	CALLS	#2, NEXT_HEADER	
		52		50	DD	00192	MOVL	RO, NEW_HEADER	
				31	13	00195	BEQL	24\$	0611
		54		52	DD	00197	MOVL	NEW_HEADER, HEADER	0612
11		56		06	E1	0019A	BBC	#6, FUNCTION, 23\$	0613
			02	A4	9F	0019E	PUSHAB	2(HEADER)	0614
	0000G	CF		01	FB	001A1	CALLS	#1, SEARCH_FCB	
				50	D5	001A6	TSTL	RO	
				05	13	001A8	BEQL	23\$	
			0800	8F	BF	001AA	CHMU	#2048	0615
					04	001AE	RET		
7E		56	01	06	EF	001AF	EXTZV	#6, #1, FUNCTION, -(SP)	0616
				53	DD	001B4	PUSHL	FCB	
				54	DD	001B6	PUSHL	HEADER	
				03	DD	001B8	PUSHL	#3	
				5E	DD	001BA	PUSHL	SP	
			0000G	CF	9F	001BC	PUSHAB	MAKE_EXTFCB	
	69			06	FB	001C0	CALLS	#6, SYSSCMKRN	
	53			50	DD	001C3	MOVL	RO, FCB	
				C1	11	001C6	BRB	21\$	0608
	6A			53	D1	001C8	CMPL	FCB, PRIMARY_FCB	0619
				1C	13	001CB	BEQL	25\$	
	53			6A	DD	001CD	MOVL	PRIMARY_FCB, FCB	0622
				53	DD	001D0	PUSHL	FCB	0623
				7E	D4	001D2	CLRL	-(SP)	
	0000G	CF		02	FB	001D4	CALLS	#2, READ_HEADER	
		54		50	DD	001D9	MOVL	RO, HEADER	
				54	DD	001DC	PUSHL	HEADER	0624
				01	DD	001DE	PUSHL	#1	
				5E	DD	001E0	PUSHL	SP	
			0000G	CF	9F	001E2	PUSHAB	UPDATE_FCB	
	69			04	FB	001E6	CALLS	#4, SYSSCMKRN	
	05		32	A5	B1	001E9	CMPL	50(PACKET), #5	0631
				29	1B	001ED	BLEQU	28\$	
				6B	D5	001EF	TSTL	CURRENT_WINDOW	0634
				08	12	001F1	BNEQ	26\$	
				53	DD	001F3	PUSHL	FCB	0635
				54	DD	001F5	PUSHL	HEADER	
				04	DD	001F7	PUSHL	#4	
	0000G	CF		03	FB	001F9	CALLS	#3, CHECK_PROTECT	
			0090	8F	BB	001FE	PUSHR	#^M<R4,R7\$	0636
				02	DD	00202	PUSHL	#2	
				5E	DD	00204	PUSHL	SP	
			0000G	CF	9F	00206	PUSHAB	READ_ATTRIB	
	69			05	FB	0020A	CALLS	#5, SYSSCMKRN	
	03			50	E8	0020D	BLBS	RO, 27\$	
				00	BF	00210	CHMU	#0	
					04	00212	RET		
		54	0000G	CF	DD	00213	MOVL	FILE_HEADER, HEADER	0637
24		56		06	E1	00218	BBC	#6, FUNCTION, 30\$	0642

ACCESS
V04-000

J 3
16-Sep-1984 00:46:05
14-Sep-1984 12:29:18

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]ACCESS.B32;1

Page 11
(2)

1C	OB	51	6B	D0	0021C	MOVL	CURRENT_WINDOW, R1	:	0644
		A1	06	E1	0021F	BBC	#6, 11(R1), 30\$:	
		50	6A	D0	00224	MOVL	PRIMARY_FCB, R0	:	0646
			A0	D5	00227	TSTL	12(R0)	:	
			07	13	0022A	BEQL	29\$:	
	0000G	CF	00	FB	0022C	CALLS	#0, REMAP_FILE	:	0647
			0D	11	00231	BRB	30\$:	
			51	DD	00233	PUSHL	R1	:	0648
			01	DD	00235	PUSHL	#1	:	
			5E	DD	00237	PUSHL	SP	:	
			CF	9F	00239	PUSHAB	MARK_COMPLETE	:	
	69		04	FB	0023D	CALLS	#4, SYSSCMKRNL	:	
	50		01	D0	00240	MOVL	#1, R0	:	0650
			04	D0	00243	RET		:	0652

; Routine Size: 580 bytes. Routine Base: \$CODE\$ + 0000

ACP
V04

```
0653 1 ROUTINE CHECK_FIND (ABD, FIB) =
0654 1
0655 1 ++
0656 1
0657 1 FUNCTIONAL DESCRIPTION:
0658 1
0659 1 This routine calls the directory search and intercepts any error
0660 1 exits to handle the create if non-existent function. If the search
0661 1 is successful, the routine returns success; if the search fails with
0662 1 no such file and the create subfunction bit is set, it returns failure;
0663 1 all other errors are resigaled.
0664 1
0665 1
0666 1 CALLING SEQUENCE:
0667 1 CHECK_FIND (ARG1, ARG2)
0668 1
0669 1 INPUT PARAMETERS:
0670 1 ARG1: address of buffer descriptor packet
0671 1 ARG2: address of FIB
0672 1
0673 1 IMPLICIT INPUTS:
0674 1 NONE
0675 1
0676 1 OUTPUT PARAMETERS:
0677 1 NONE
0678 1
0679 1 IMPLICIT OUTPUTS:
0680 1 NONE
0681 1
0682 1 ROUTINE VALUE:
0683 1 1 if find is successful
0684 1 0 if file is to be created
0685 1
0686 1 SIDE EFFECTS:
0687 1 NONE
0688 1
0689 1 --
0690 1
0691 2 BEGIN
0692 2
0693 2 MAP
0694 2 ABD : REF BBLOCKVECTOR [,ABD$C_LENGTH],
0695 2 FIB : REF BBLOCK;
0696 2
0697 2 BUILTIN
0698 2 FP;
0699 2
0700 2 EXTERNAL ROUTINE
0701 2 FIND; ! find file in directory
0702 2
0703 2
0704 2 ! Establish the condition handler and call FIND. If we hear from it we
0705 2 return true. Any signals cause either unwind or resigal.
0706 2
0707 2
0708 2 .FP = HANDLER;
0709 2
```


ACCESS
V04-000

L 3
16-Sep-1984 00:46:05
14-Sep-1984 12:29:18

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]ACCESS.B32;1
Page 13
(3)

```
.. 398      0710 2 FIND (.ABD, .FIB, 0);  
.. 399      0711 2 RETURN 1;  
.. 400      0712 2  
.. 401      0713 1 END;
```

! end of routine CHECK_FIND

0000 00000 CHECK_FIND:									
	6D	0000V	CF	9E	00002	.WORD	Save nothing	..	0653
			7E	D4	00007	MOVAB	HANDLER, (FP)	..	0708
	7E	04	AC	7D	00009	CLRL	-(SP)	..	0710
0000G	CF		03	FB	0000D	MOVQ	ABD, -(SP)	..	
	50		01	D0	00012	CALLS	#3, FIND	..	0711
				04	00015	MOVL	#1, R0	..	0713
						RET		..	

; Routine Size: 22 bytes, Routine Base: \$CODE\$ + 0244

```
403 0714 1 ROUTINE HANDLER (SIGNAL, MECHANISM) =
404 0715 1
405 0716 1 ++
406 0717 1
407 0718 1 FUNCTIONAL DESCRIPTION:
408 0719 1
409 0720 1 This routine is the condition handler for the conditional find call.
410 0721 1 It intercepts the error exit from FIND and unwinds to CHECK_FIND's
411 0722 1 caller when appropriate.
412 0723 1
413 0724 1
414 0725 1 CALLING SEQUENCE:
415 0726 1 HANDLER (ARG1, ARG2)
416 0727 1
417 0728 1 INPUT PARAMETERS:
418 0729 1 ARG1: address of signal array
419 0730 1 ARG2: address of mechanism array
420 0731 1
421 0732 1 IMPLICIT INPUTS:
422 0733 1 NONE
423 0734 1
424 0735 1 OUTPUT PARAMETERS:
425 0736 1 NONE
426 0737 1
427 0738 1 IMPLICIT OUTPUTS:
428 0739 1 NONE
429 0740 1
430 0741 1 ROUTINE VALUE:
431 0742 1 $$$_RESIGNAL or none if unwind
432 0743 1
433 0744 1 SIDE EFFECTS:
434 0745 1 NONE
435 0746 1
436 0747 1 --
437 0748 1
438 0749 1
439 0750 2 BEGIN
440 0751 2
441 0752 2 MAP
442 0753 2 SIGNAL : REF BBLOCK, ! signal arg array
443 0754 2 MECHANISM : REF BBLOCK; ! mechanism arg array
444 0755 2
445 0756 2 EXTERNAL ROUTINE
446 0757 2 $$$UNWIND : ADDRESSING_MODE (ABSOLUTE);
447 0758 2 ! system unwind service
448 0759 2
449 0760 2
450 0761 2 ! If the condition is change mode to user (error exit) and the status is
451 0762 2 ! no such file, cause an unwind to return 0 to the access main line.
452 0763 2 ! Otherwise, just resignal the condition.
453 0764 2 !
454 0765 2
455 0766 2 IF .SIGNAL[CHFSL_SIG_NAME] EQL $$$CMODUSER
456 0767 2 AND .SIGNAL[CHFSL_SIG_ARG1] EQL $$$NOSUCHFILE
457 0768 2 THEN
458 0769 3 BEGIN
459 0770 3 MECHANISM[CHFSL_MCH_SAVRO] = 0;
```

ACCESS
VG4-000

N 3
16-Sep-1984 00:46:05
14-Sep-1984 12:29:18

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]ACCESS.B32;1 Page 15
(4)

```
: 460      0771 3  SYSSUNWIND (0, 0);
: 461      0772 2  END;
: 462      0773 2
: 463      0774 2  RETURN SSS_RESIGNAL;
: 464      0775 2
: 465      0776 1  END;
```

! status is irrelevant if unwinding
! end of routine HANDLER

.EXTRN SYSSUNWIND

```
0000 00000 HANDLER: .WORD Save nothing      : 0714
                                MOVL SIGNAL, R0      : 0766
00000424 50      04 AC D0 00002
                                CMPL 4(R0), #1060
                                BNEQ 1$
00000910 8F      08 AO D1 00006
                                CMPL 8(R0), #2320      : 0767
                                BNEQ 1$
                                MOVL MECHANISM, R0      : 0770
                                CLRL 12(R0)
                                CLRG -(SP)      : 0771
00000000G 9F      0C AO D4 0001E
                                CALLS #2, @SYSSUNWIND
                                MOVZWL #2328, R0      : 0774
                                RET      : 0776
```

; Routine Size: 48 bytes, Routine Base: \$CODE\$ + 025A

```
: 466      0777 1
: 467      0778 1  END
: 468      0779 0  ELUDOM
```

PSECT SUMMARY

Name	Bytes	Attributes
\$CODE\$	650	NOVEC,NOWRT, RD , EXE,NOSHR, LCL, REL, CON,NOPIC,ALIGN(2)

Library Statistics

File	----- Total	Symbols Loaded	----- Percent	Pages Mapped	Processing Time
_\$255\$DUA28:[SYSLIB]LIB.L32;1	18619	50	0	1000	00:01.9

ACCESS
V04-000

B 4
16-Sep-1984 00:46:05
14-Sep-1984 12:29:18

VAX-11 Bliss-32 V4.0-742
DISK\$VMSMASTER:[F11A.SRC]ACCESS.B32;1 Page 16
(4)

COMMAND QUALIFIERS

: BLISS/CHECK=(FIELD,INITIAL,OPTIMIZE)/LIS=LIS\$:ACCESS/OBJ=OBJ\$:ACCESS MSRC\$:ACCESS/UPDATE=(ENHS\$:ACCESS)

: Size: 650 code + 0 data bytes
: Run Time: 00:17.3
: Elapsed Time: 00:45.0
: Lines/CPU Min: 2697
: Lexemes/CPU-Min: 15424
: Memory Used: 226 pages
: Compilation Complete

0164 AH-BT13A-SE
VAX/VMS V4.0

DIGITAL EQUIPMENT CORPORATION
CONFIDENTIAL AND PROPRIETARY